Static meta-level architecture of Tearless

Transforming JavaScript callbacks into continuation-passing style

Joeri De Koster, Laure Phillips

VUB VRIJE UNIVERSITEIT BRUSSEL

KATHOLIEKE UNIVERSITEIT LEUVEN

AGENTSCHAP INNOVEREN & ONDERNEMEN

tierless programming

enabling technologies

libraries

tools

Deliverable 1.1.1
Static MLA

Static Meta-Level Architecture

Dynamic Meta-Level Architecture

tier-splitting transpiler

consistent replicated state

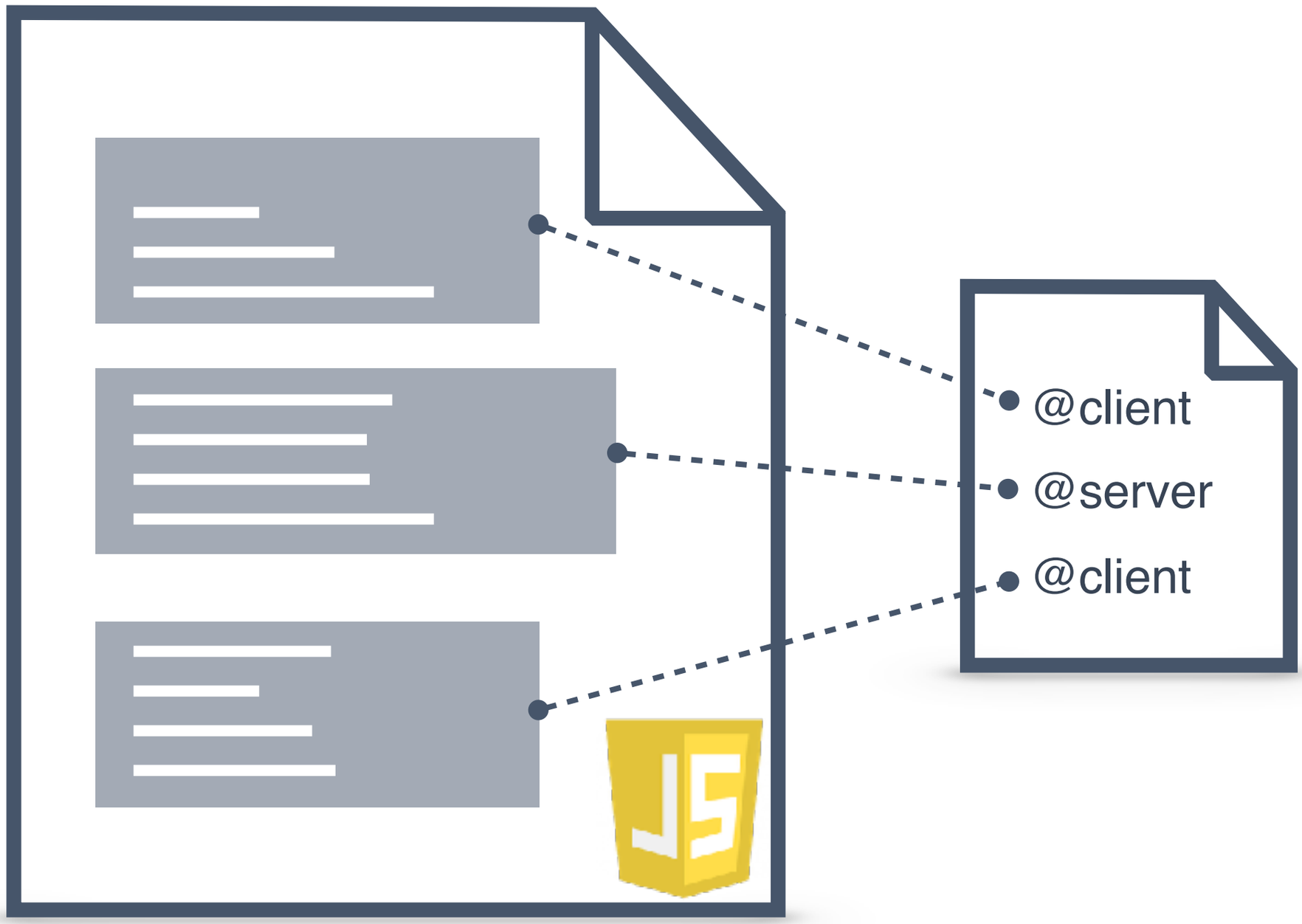
end-to-end security monitor

end-to-end debugger

Deliverable 1.2.1
Simple Transpiler

migration assistent

Deliverable 2.1.1
Data Model Slicing

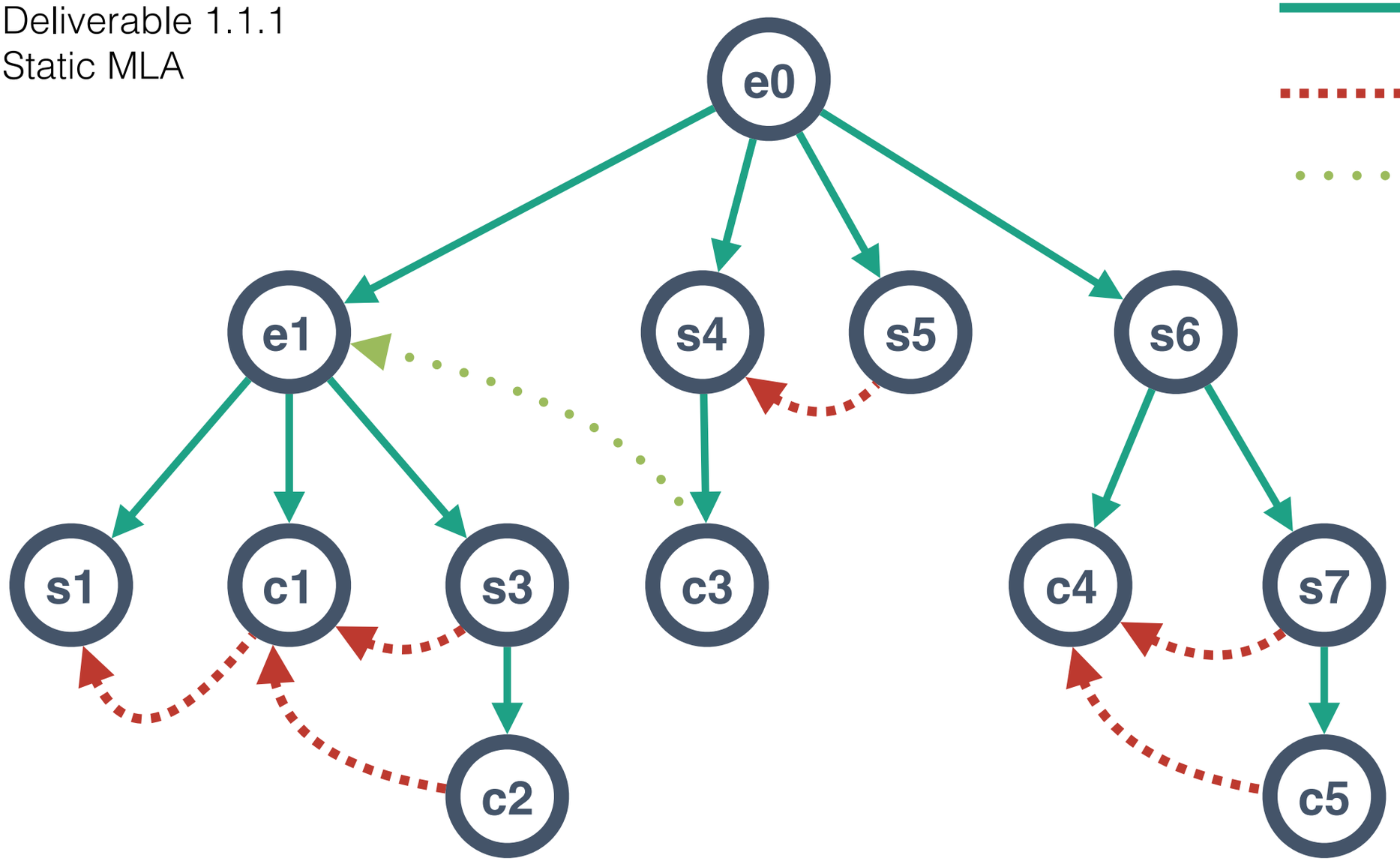

@client
@server
@client

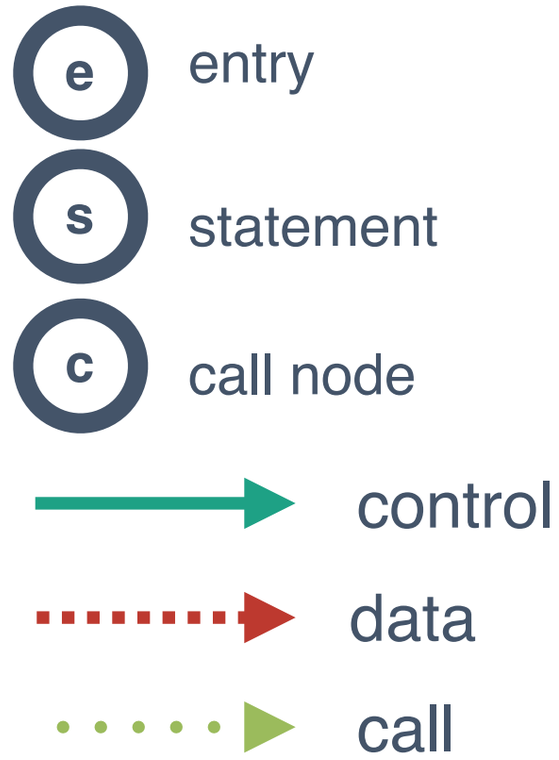
```
1  /* @server */
2  {
3    function broadcast(msg) {
4      displayMessage(msg);
5    }
6  }
7
8  /* @client */
9  {
10   var clientId = "user" + Math.random();
11
12   function displayMessage(msg) {
13     $("#msgs").append(msg);
14   }
15
16   broadcast(clientId + " says hello!");
17
18 }
```

CPS Transformation

Based on Program Dependency Graph

Deliverable 1.1.1
Static MLA





```
/* @server */
{
  function broadcast(msg) {
    displayMessage(msg);
  }
}

/* @client */
{
  var clientId = "user" + Math.random();

  function displayMessage(msg) {
    $("#msgs").append(msg);
  }

  broadcast(clientId + " says hello!");

}
```

Tiersplit by program slicing

```
/* SERVER */
server.expose({
  'broadcast': function (msg, callback) {
    server.rpc("displayMessage", [msg]);
  }
});

/* CLIENT */
var clientId;

clientId = "user" + Math.random();

client.rpcCall('broadcast', clientId + ' says hello!', function (err0, res0) {
});

client.expose({
  'displayMessage': function (msg, callback) {
    $(‘#msgs').append(msg);
  }
}
```

Deliverable 1.2.1
Simple Transpiler

Inferring Annotations

```
1 /* @server */
2 {
3   /* @remoteFunction */
4   function broadcast(msg) {
5     /* @remoteCall */
6     displayMessage(msg);
7   }
8 }
```

without analysis

with analysis

```
1 /* @server */
2 {
3   function broadcast(msg) {
4     displayMessage(msg);
5   }
6 }
```

Independent calls executed concurrently

```
1 /* @client */
2 {
3   function requestServices() {
4     requestOne();
5     requestTwo();
6   }
7 }
```

without analysis

conservative

```
1 /* CLIENT */
2 {
3   function requestServices() {
4     client.rpcCall(“requestOne”, function (err1, res1) {
5       client.rpcCall(“requestTwo”, function (err2, res2) {
6       }
7     }
8   }
9 }
```

Independent calls executed concurrently

```
1 /* @client */
2 {
3   function requestServices() {
4     requestOne();
5     requestTwo();
6   }
7 }
```


```
1 /* CLIENT */
2 {
3   function requestServices() {
4     client.rpcCall(“requestOne”, function (err1, res1) {
5     }
6     client.rpcCall(“requestTwo”, function (err2, res2) {
7     }
8   }
9 }
```

Tierless Annotations

PLACEMENT	COMMUNICATION	DATA SHARING	FAILURE HANDLING
@	@	@	@
client	remoteCall	local	defineHandler
server	remoteFunction	copy	useHandler
shared	reply	observable	
ui	broadcast	replicated	
	blocking		

Deliverable 1.2.1
Simple Transpiler

Deliverable 2.1.1
Data Model Slicing

What this means for you now

```
1 fs.readdir(source, (err, files) => {
2   for(file of files) {
3     fs.readFile(file, (err, data) => {
4       // do something with file
5     });
6   };
7 });
```

Callback Hell

```
1 var files = fs.readdir(source);
2 for(file of files) {
3   var data = fs.readFile(file);
4   // do something with file
5 }
```

Dependence-Driven Delimited CPS Transformation for JavaScript
Laure Philips, Joeri De Koster, Wolfgang De Meuter and Coen De Roover

CPS Transformation Tools for JS

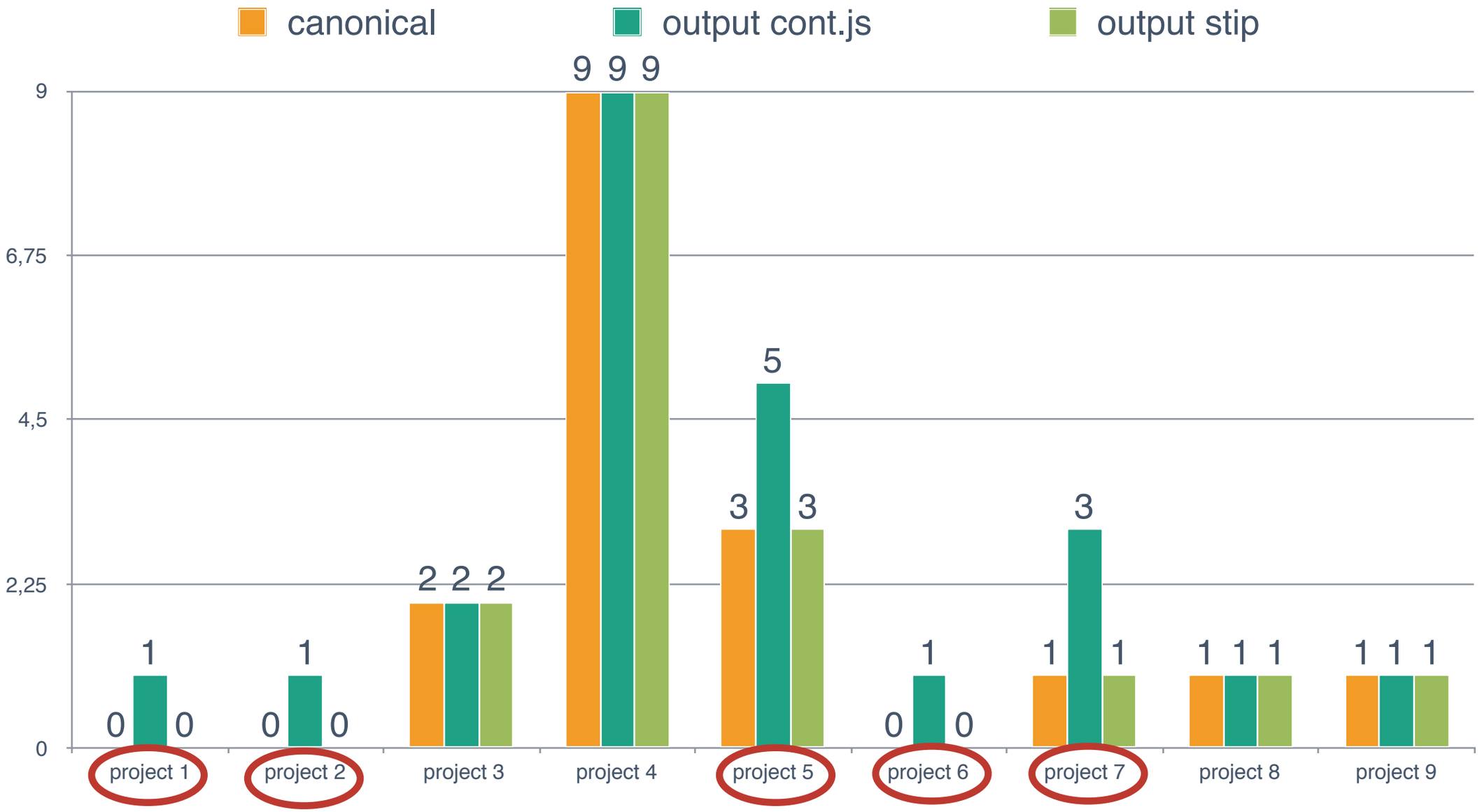
	streamline.js	continuation.js	tameJS	stratifiedJS	stip.js
Annotations	_ construct	cont(params)	defer(params)	resume	optional
Failure handling	✓	✓	✓	✓	✓
Readable output	✗	✓	✗	✗	✓
Tool support	✓	✗	✓	✓	✓✗
Minimal nesting of CB	✗	✗	✗	✗	✓

Quantitative Evaluation

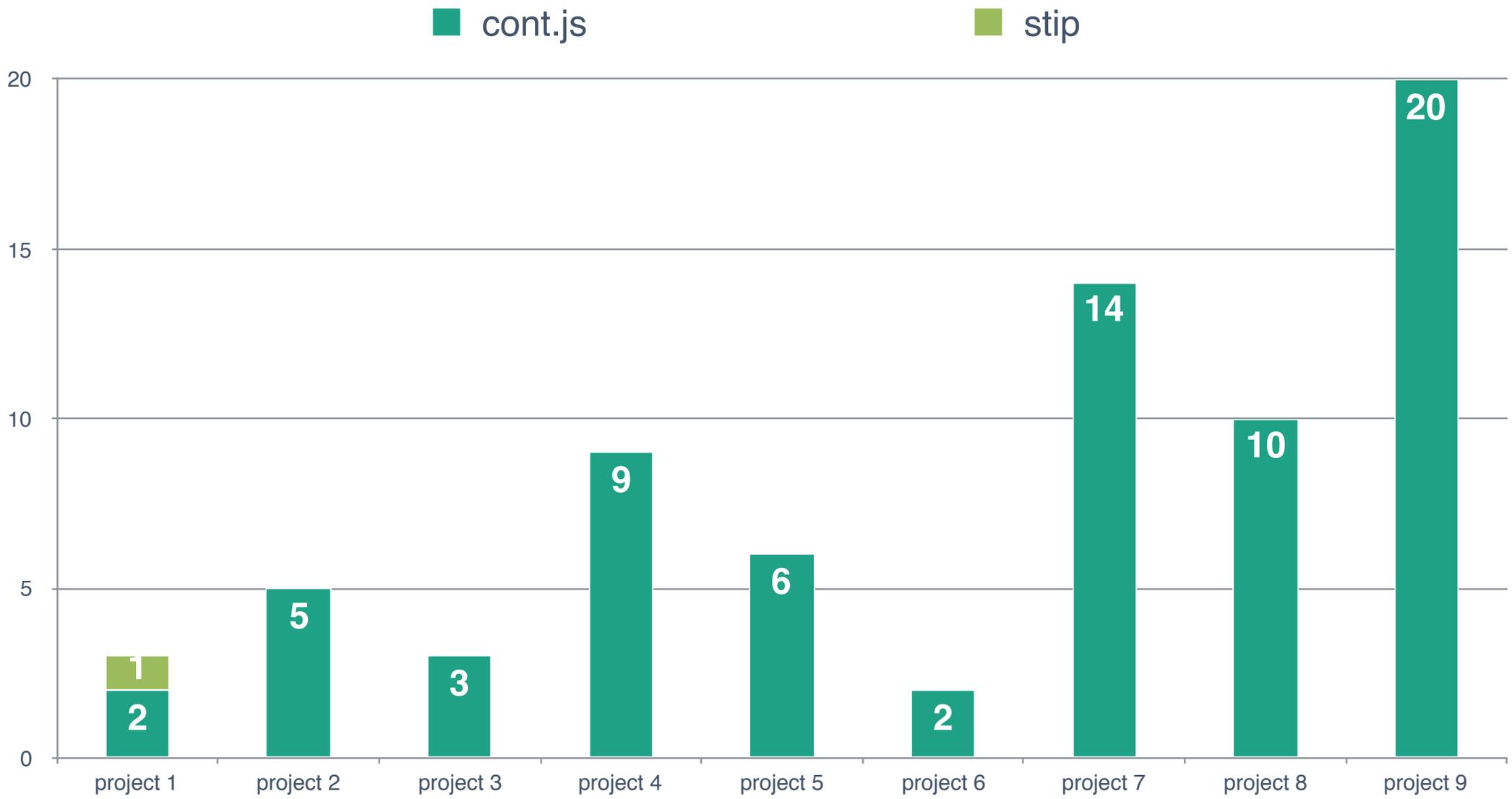
project	loc	# callbacks	max nesting level	average # statements in callback
1	35	2	0	2.5
2	13	2	0	1
3	28	4	2	2
4	54	10	9	1
5	65	6	3	2.3
6	29	2	0	1
7	111	14	1	1.1
8	146	10	1	1.1
9	519	21	1	1.7

examples taken from callbackhell.com

Maximal callback nesting level

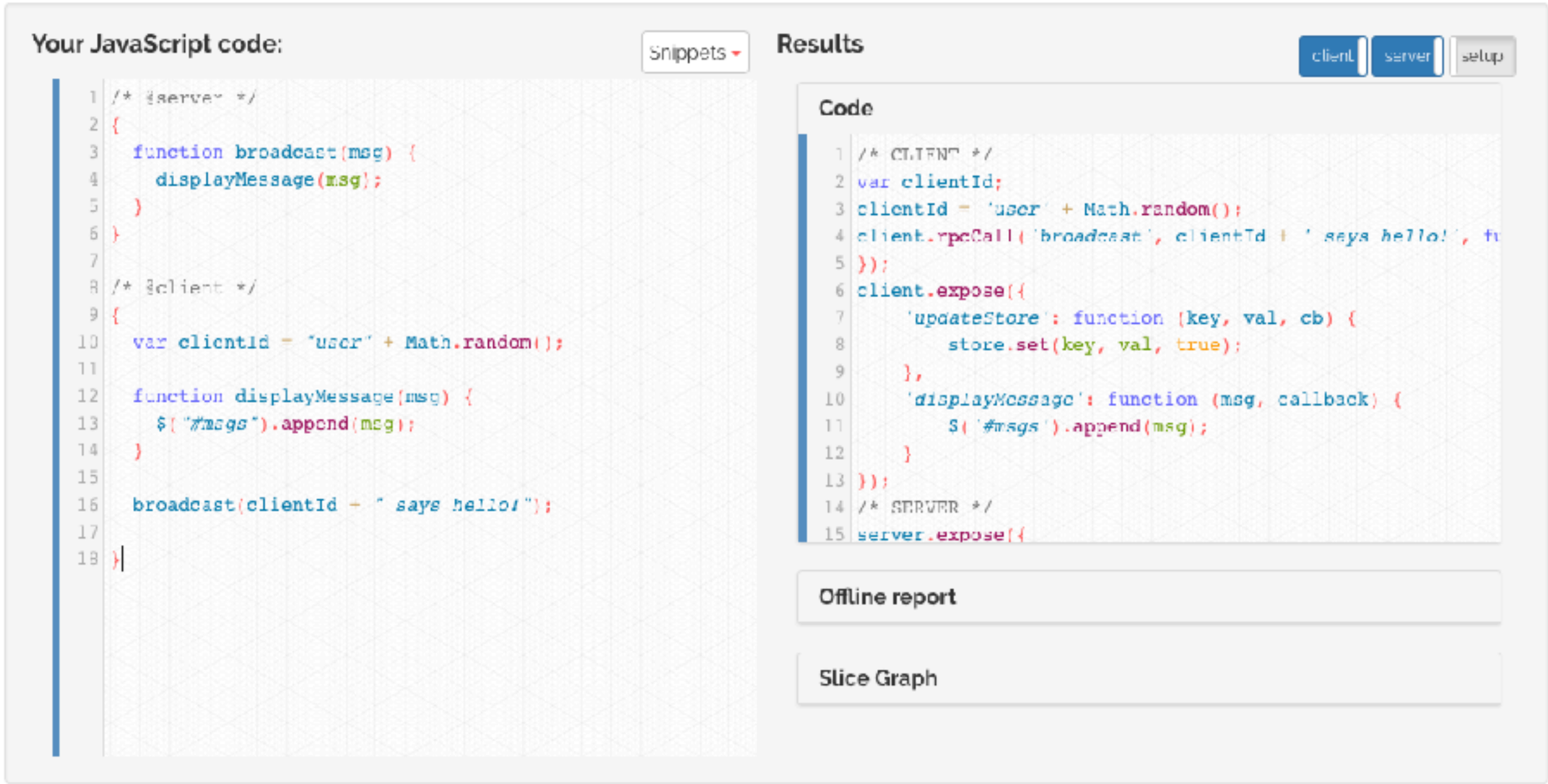


Developer hints required



Help Wanted!

bit.ly/stipjs



Summary

- Short term
  - CPS transformation  tool
  - Replicated State  library

- Long Term
  - Tierless Framework